Technical Report: AIC-01-001, Naval Research Laboratory, Code 5515
4555 Overlook Avenue SW, Washington, DC, 20375-5537, January, 2001.

Summary Report: Design for Safety Program Project
Integrated Lessons Learned Systems
Duration: 1 August – 30 November 2000

David W. Aha
Head, Intelligent Decision Aids Group
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory (Code 5515)
4555 Overlook Ave., SW
Washington, DC 20375
aha@aic.nrl.navy.mil (202) 404-4940 | (202) 767-3172 (FAX)

Kalyan Moy Gupta
ITT Industries
2560 Huntington Ave.
Alexandria, VA 22303
gupta@aic.nrl.navy.mil (202) 767-6645 | (202) 767-3172

1. Objective

This summary report summarizes a four-month project funded by the Design for Safety Program (POC: H. Stewart, Knowledge Engineering). The project leader and his colleagues who worked on this effort are located in Washington, DC. Contact with members of the DFS Program was maintained by weekly phone conversations, frequent email communication, and attendance at the DFS Workshop (DFS2000), which took place during 10-12 October 2000.

1.1 Background

This project concerns knowledge management (KM) objectives within the DFS Program (www.dfs.nasa.gov), which identified KM as one of three broad problem classes of NASA programs that requires attention. More specifically, DFS identified that program success depends heavily on the expertise of project individuals (e.g., knowledge, interactions), but that this expertise is often not being captured. DFS goals to address this problem include developing NASA-wide solutions for mapping/capturing this knowledge into databases that can subsequently be mined by other NASA employees/contractors. Importantly, mining should not be passive (i.e., a *pull* process), but instead be a proactive *push* process in which relevant recorded expertise is made available to project individuals at the time they require it, and in their working (e.g., software) environment.

Several NASA programs, including those that maintain Problem Reporting and Corrective Action (PRACA) systems, share this goal. As explained in the DFS *PRACA Enhancement Pilot Study Report*, which focused on assessing the PRACA systems of the Space Shuttle Program (SSP), these systems are currently a diverse collection of computer hardware, software, networks, and databases, and paper files distributed across several NASA, USA and other contract support sites. Highly trained domain experts are devoted to maintaining these resources, and most of the systems are accessible for viewing using the ADAM centralized data warehouse. Unfortunately, ADAM's contents cannot be mined or navigated easily by non-expert users, which requires access to the domain experts themselves because their PRACA usage knowledge is not captured or documented. This is a key reason why the Pilot Study's authors conclude that PRACA is incapable of supporting program-level risk assessment, and will not scale well.

NASA is addressing these problems in part by supporting on-line *lessons learned* repositories, among which the Lessons Learned Information System (LLIS) is most prominent (llis.nasa.gov). Although potentially useful, standalone lesson retrieval tools, such as the two maintained by LLIS (and the dozens maintained in other government, military, and industrial organizations), have a dreadful reputation and are generally ignored by their customers (i.e., in this case, project personnel who can benefit from the lessons) (Weber et al., 2001). Complaints range from concerns that the lesson content fluctuates wildly in utility to unsolved social and technical issues (e.g., lack of time or skill to fetch lessons, impoverished search capabilities), and redesigning the LLIS is one of three primary thrusts of the NASA CIO KM Pilot Project. One possible solution is to use portals and spiders to proactively search and for retrieve relevant documents for project personnel, but they don't address issues of knowledge capture and representation, and are by nature standalone solutions.

In contrast, we advocate the development of lessons management tools that, through tight integrations with lesson-targeted decision support tools, can proactively prompt users with relevant lessons at the time they can be used (Weber et al., 2000). To implement this approach at NASA requires (1) an environment to capture lessons, (2) processes for verifying/validating lesson content, and (3) a decision support tool in which to distribute lessons. The first of these goals was the focus of this project.

1.2 Deliverables

Our primary deliverables for this project are an architecture for lesson management, named LMTS, a representation for lessons, and a lesson editor, capable of being integrated with the Common Object Relationship Engine (CORE) (Knight & Aha, 2000). We will discuss our work's relation to CORE and Postdoc in Section 2.

1.2.1 Lesson Management Tool Set (LMTS) Architecture

Our goal was to develop an architecture with the following characteristics:

?? It should support a full range of intelligent lessons management processes, from collection through distribution.
?? It should be easily integratable with a variety of lesson management tools.
?? It should be easily extensible, allowing users to add functionality so that the system can evolve over time to meet user needs.

Figure 1 displays the architecture we conceived for lesson management based on this specification. The tools for lesson collection, shown in orange background, include an interactive lesson elicitor (Lesson Editor), a tool for monitoring emails to identify potential lessons (Source Monitoring Agent), a lesson extraction tool (Lesson Miner), and a tool for requesting an email author to submit a lesson (Author Notification Agent). Figure 2 then displays how these tools interact. Briefly, a user interacts with them by

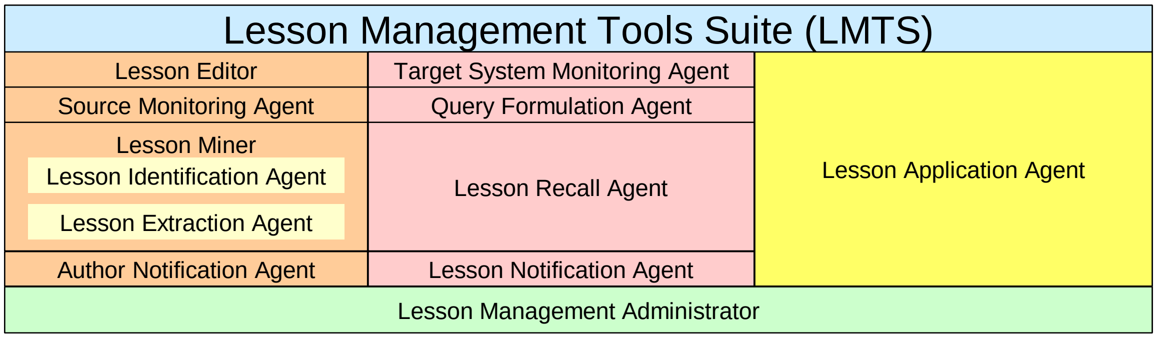

Lesson Management Tools Suite (LMTS)
Lesson Editor
Source Monitoring Agent
Lesson Miner
Lesson Identification Agent
Lesson Extraction Agent
Author Notification Agent
Target System Monitoring Agent
Query Formulation Agent
Lesson Recall Agent
Lesson Notification Agent
Lesson Application Agent
Lesson Management Administrator

Figure 1: Architecture for Lesson Management.

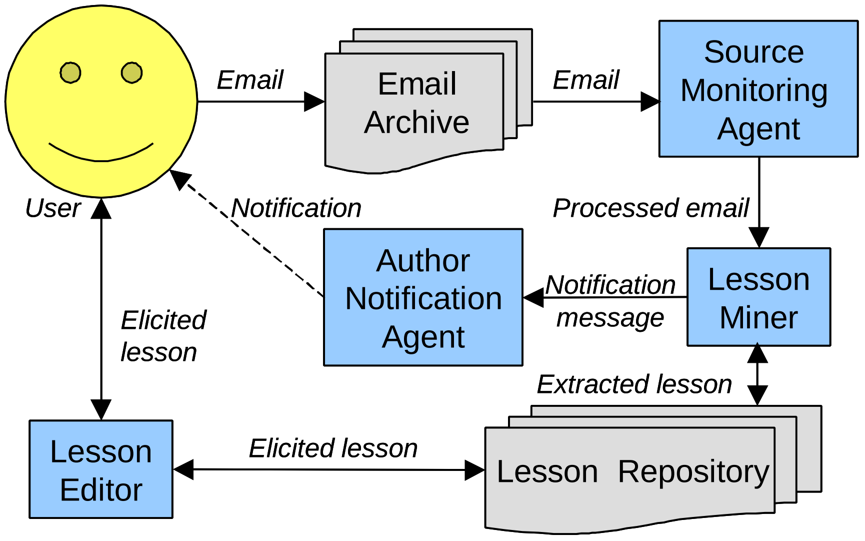


Figure 2: Interaction among the Lesson Collection Tools.

first submitting an email to Postdoc's email archive. At that time, the Source Monitoring Agent processes the email so that the Lesson Miner can determine, with high probability, whether the email contains a lesson that should be stored. If so, it wakes up the Author Notification Agent, which would email a request to that author, asking them to use the Lesson Editor to elicit their lesson into the Lesson Repository.

The components in pink in Figure 1 concern lesson distribution. The intent is that they will be tightly integrated with a Lesson Application Agent, which is used by NASA project personnel. The Target System Monitoring Agent's job is to monitor the Lesson Application Agent, checking for user interface events. It passes them to the Query Formulation Agent, which interprets these events and translates them into a query to the Lesson Recall Agent. This, in turn, tests whether the query closely matches the conditions of one of the rules (See Section 1.2.2 for discussion on lesson representation, including lesson conditions). If so, then it retrieves those lessons and activates the Lesson Notification Agent, which will prompt the user accordingly.

The job of the **Lesson Management Administrator** ensures that all these tools are configured so that they can interact. For example, the Target System Monitoring Agent must be made aware of the user's interface events with the Lesson Application Agent (e.g., a decision support tool).

Together, the tools of this architecture cooperate to achieve the objectives of intelligent lesson management, as listed above. We discuss their implementation status in Section 1.2.3.

1.2.2 Lesson Representation

In choosing a representation for lessons, we examined other lessons learned efforts at NASA (e.g., LLIS, RECALL), and concluded that a more expressive representation than is currently used would be required to obtain the level of detail necessary for reasoning with the lesson (i.e., to identify project situations in which it could be relevant). With this in mind, we developed the following representation after discussing these issues with a few NASA representatives (e.g., Kanna Rajan, Code 269):

1. **Identification**: Id, Title, Contributors, Project Information
2. **Observations**: Summary, Detailed List
3. **Analysis**: Cause, Effect, Notes
4. **Recommendation**: Recommended Corrective Action, Benefits Analysis, References
5. **Administration**: Edit History, Edit Comment Records, Administrative Information

Some of these items are simple (e.g., the Title) while others are complex (e.g., Project Information is decomposed into Project, Task, Deliverables, Resources, and Roles). Figure 2 displays a screenshot from the current software prototype, in which the Deliverables of the Project Information subfield of the Identification field are being entered. The tree shown on the right of this figure displays the possible deliverables that can be inserted into the Deliverables subfield. In general, we use simple taxonomies to control the selection of entries into each list field. In this way, the entries into a lesson's list fields can be tracked to its categorization in the list field's taxonomy. For example, the names appearing in the list of Contributors are obtained only from the Contributors taxonomy, which will display the organizational hierarchies for the individuals listed. The interface has been carefully designed around this concept of taxonomic entry, such that it is easy to re-order the entries in a list, to add and delete entries, and to modify the taxonomies from which they were selected.

1.2.3 Lesson Editor

In this first phase of a larger project, we have now implemented the Lesson Editor, as shown in Figure 1. It has the following characteristics:

?? Supports lesson engineering workflow, from lesson entry to lesson publication.

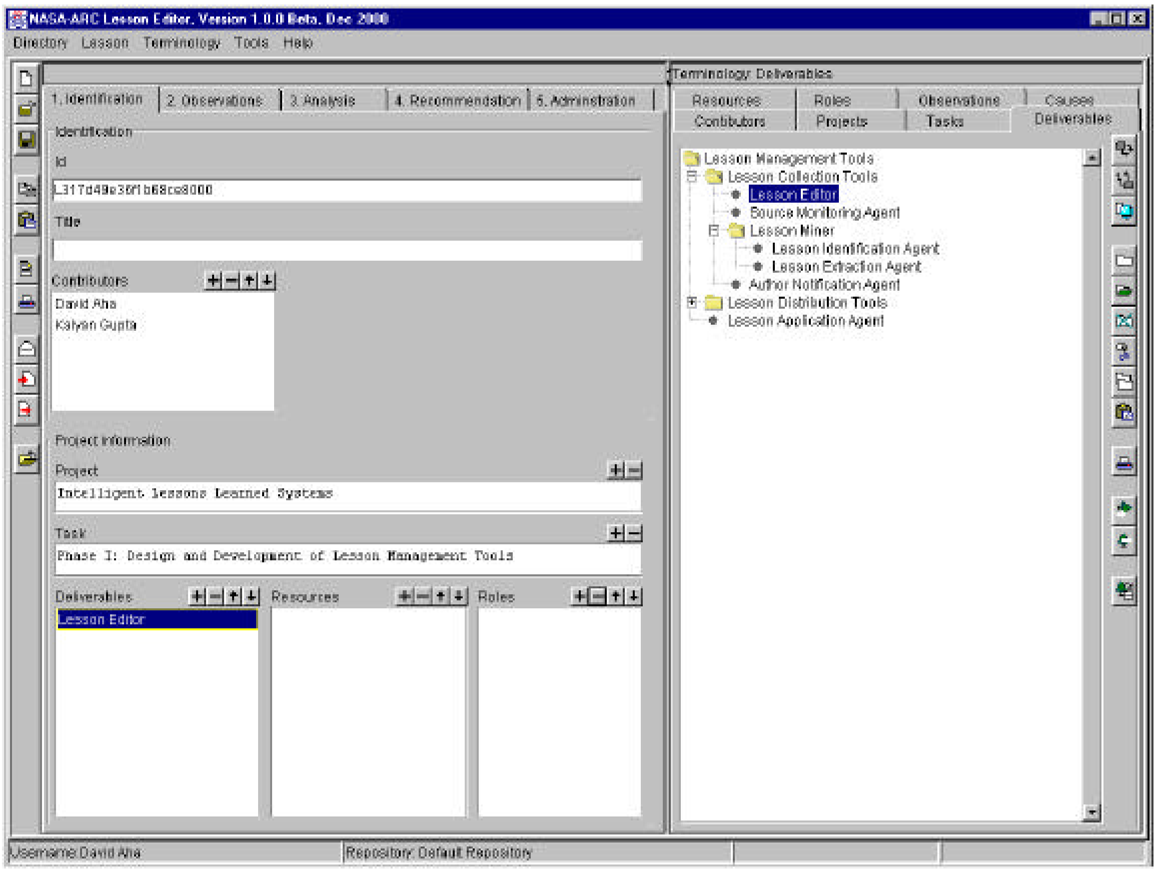
?? Supports structured lesson representation, allowing users to select lesson features from pre-established taxonomies.
?? Contains a built-in editor for creating/maintaining these taxonomies.
?? Produces XML-compliant lessons for easy exchange across systems.
?? Java 1.2 (application & applet, for use in a WWW browser)

These characteristics will allow the Lesson Editor to be easily integratable with the redesigned Postdoc environment (Knight & Aha, 2000). Authoring and publishing a lesson involves participation from various individuals (e.g., authors, reviewers) in a coordinated workflow process for publishing high-quality lessons. The Lesson Editor supports this process by simplifying the process of eliciting, reviewing, and editing high-quality lessons.

A key component of our approach is the use of taxonomies, which organize various fields that are recorded in a lesson (e.g., names of contributors, projects, tasks), to facilitate the lesson insertion process. Taxonomies are useful for the purpose of generalizing lesson contents; for a given field (e.g., names of projects), its taxonomy lists an organized representation of all known possible entries in that field. Using a taxonomy to insert field values for a lesson involves two steps: taxonomy generation and selecting items from a taxonomy. A taxonomy can be generated in several ways. However, as of now, we anticipate that it will be generated through iterative refinement of its tree structure; the user can quickly insert new nodes in this tree that denote either groups of feature values (e.g., DFS Program projects) or a specific feature value (e.g., DFS PRACA Pilot Study Project). Selecting a field value from a taxonomy is simple; the user need only to double click on the value to place that value in the lesson's target field. For example, Figure 3 displays a screenshot of the Lesson Editor for our own ILLS project in which the (simplified) taxonomy for deliverables is shown on the right-hand side and the lower left panel displays the deliverables claimed for the project (i.e., the Lesson Editor).

The reason for generating XML-compliant lessons is that XML is the emerging standard for exchanging data between systems. In addition to the lesson itself, the taxonomies are also stored in an XML file. In fact, all of the data used by the editors in our tools are in XML.

We are implementing these tools in Java 2 because it is a platform-independent programming environment. Also, through using applets, Java-encoded tools can easily be made available through WWW browsers.


NASA-ARC Lesson Editor. Version 1.0.0 Beta. Dec 2000
Directory Lesson Terminology Tools Help
1. Identification
2. Observations
3. Analysis
4. Recommendation
5. Administration
Terminology: Deliverables
Resources Roles Observations Causes
Contributors Projects Tasks Deliverables

Identification

Id: L317d49e36f1b68ce8000

Title:

Contributors: David Aha; Kalyan Gupta

Project Information

Project: Intelligent Lessons Learned Systems

Task: Phase I: Design and Development of Lesson Management Tools

Deliverables: Lesson Editor

Resources:

Roles:

```
Lesson Management Tools
  Lesson Collection Tools
    Lesson Editor
    Source Monitoring Agent
    Lesson Miner
      Lesson Identification Agent
      Lesson Extraction Agent
    Author Notification Agent
  Lesson Distribution Tools
  Lesson Application Agent
```

Username: David Aha | Repository: Default Repository

Figure 3: The Lesson Editor, during insertion of items in the Deliverables list for our project.

2. Lesson Editor

We have designed the Lesson Editor's interface so that it has the following menus:

- *Directory*: Provides access to the lesson's directory for a particular user.
- *Lesson*: Allows users to create, edit, and administer a lesson.
- *Terminology*: An intuitive synonym for "taxonomies", this provides users with the ability to maintain taxonomies, independent of their lessons.
- *Tools*: Will allow users to specify options for controlling the Lesson Editor's behavior.
- *Help*: Provides documentation.

The primary way in which users will interact with the Lesson Editor while creating lessons is shown in Figure 3. The left-hand side five lesson representation fields, and in this case shows details for the first of these fields (Identification). The right-hand side shows the seven taxonomies and list of contributors. In this figure, the Deliverables taxonomy is shown.

This editor is designed to be a lesson collection and management tool, and will be integrated with the redesigned Postdoc system Postdoc is a NASA-ARC WWW document and distributed collaboration environment that is in use by thousands of NASA employees and collaborators from education and industry. It is an effective tool for organizing, storing, and retrieving documents of many types (e.g., word processing, spreadsheet, presentation, image, video). Its redesign is being supported by DFS (POC: H. Stewart), and its planned distinguishing capabilities will include an object-oriented architecture, end-user application extension capabilities, and access control techniques. The Lesson Editor is intended to demonstrate the redesigned Postdoc's capabilities to support knowledge management extensions.

At this time, we are completing implementation of the Lesson Editor, with future goals to implement the remaining lesson collection and distribution tools, as shown in Figure 1.

3. Future Work

Our future goals are twofold. First, we will continue development of support tools for the Lessons Collection process. Afterwards, we will focus on development of support tools for Lessons Distribution in conjunction with a suitable NASA project management tool, which will interact directly with the Lesson Application Agent.

3.1 Lessons Collection

The Lessons Editor is the first of several tools required for lessons collection. As shown in Figure 2, we intend to develop a Source Monitoring Agent, a Lesson Miner, and an Author Notification Agent.

It should be clear to the reader that the amount of information being requested for entering the lesson is substantial, and this will cause some potential lesson authors to decide to not enter their lessons (due to the complexity of lesson entry). Therefore, we intend to pursue research activities to reduce the amount of effort required to enter lessons using this approach. For example, we intend to use a *conversational case retriever* (Aha & Breslow, 1997) to assess a new lesson's context, which we then intend to use to automatically complete parts of the lesson. For example, if the user can answer a few questions that identify the project and process being targeted by the lesson, than several fields can be filled in using default information stored in the taxonomies. We also intend to use this approach to assist with defining the taxonomies, and intend to pursue alternative textual data mining approaches to also assist with taxonomy extraction. Combined, these tools could greatly simplify the complexity of lesson entry.

3.2 Lessons Distribution

We intend to support both (1) standalone[1] lesson distribution and (2) integrated lesson distribution management tools. Standalone distribution will be supported using a suitable retrieval tool, perhaps beginning with the functionalities supported for retrieving LLIS

[1] By "standalone", we mean an independent lesson distribution system, not integrated with other lesson management software.

lessons and then extending them to exploit the more expressive representation for lessons used here.

Lesson distribution in the context of project management tools is the ultimate goal of our NASA project. We have previously demonstrated how proactive lesson distribution can be implemented in the context of a military plan authoring tool suite (Weber et al., 2000), and we will use this framework as a starting point for designing a similar functionality for NASA project management tools. In this context, we are aware of two options for targeting our efforts. First, STIN is a NASA-Marshall project management and progress-reporting tool that may be ideal for evaluating our approach. We intend to discuss this idea further with both DFS Program Managers and STIN POCs (e.g., Steve Cook). Second, we wish to identify risk assessment tools that could benefit from an integrated lesson distribution capability. In either case, our goal is to design an experiment to evaluate the utility of integrated lesson distribution.

Currently, search for lessons is done using a keyword search approach. We believe that the taxonomies being developed for lesson collection can be used as the basis of a superior search mechanism. For example, if the user wishes to locate lessons concerning a specific program, they can use the Project taxonomy to reduce their search to projects in the specified program or projects that are related (i.e., have characteristics that are similar to projects in the specified program). This capability does not currently exist in the LLIS lesson retrieval tool, which is restricted to exact matching of index terms.

3.2 Evaluation

Both our lesson collection and, eventually, lesson distribution tools must undergo user testing to determine their suitability for deployment at NASA centers. As we develop our understanding of NASA's working culture and related efforts on knowledge management, we will be in a better position to identify the types of user testing that are needed, which may be based on observations of user testing methodologies that have been previously, and successfully, used to evaluate other software systems. Prior to doing user testing, we anticipate conducting simulated user testing in our laboratory, in which rigorous empirical studies will be conducted to, for example, evaluate the capability of the Lesson Miner's accuracy in identifying lessons in user emails.

4. Conclusion

In conclusion, in addition to identifying and interviewing key groups in NASA that are contributing to PRACA, KM, and lessons learned efforts, we are completing a first step towards providing lessons learned management functionality in the redesigned Postdoc WWW document management system. We have created a Lessons Editor, which involved selecting a representation for lessons, a suitable interface for lesson entry, and an approach for representing background knowledge (i.e., in taxonomies). This editor can be used to elicit project-related lessons, both technical and managerial, that can then be used to assist in subsequent project decision-making. In this way, we intend to provide tools that support a proactive knowledge management strategy for capturing and sharing project information for NASA employees and contractors.

References

Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference in Case-Based Reasoning*. Providence, RI: Springer-Verlag.

Knight, C., & Aha, D.W. (2000). A common knowledge framework and lessons learned module. In D.W. Aha & R. Weber (Eds.) *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop* (Technical Report WS-00-03). Menlo Park, CA: AAAI Press.

Weber, R., Aha, D.W., & Becerra-Fernandez, I. (2001). Intelligent lessons learned systems. To appear in *Expert Systems with Applications*.

Weber, R., Aha, D. W., Muñoz-Avila, H., & Breslow, L.A. (2000). Active delivery for lessons learned systems. *Proceedings of the Fifth European Workshop on Case-Based Reasoning* (pp. 322-334). Trento, Italy: Springer.